Quantil

Technical Validation Note

Testing the Monte Carlo engine against an independent Python reimplementation

Cost & Schedule Risk Analyzer for Excel

Version 1.1, October 2026

Subject	Calculation engine of the probabilistic risk assessment (Monte Carlo engine)
Component	modMonteCarlo (engine v1.0): Quantil Cost & Schedule Risk Analyzer for Excel (v1.1)
Method	Beta-PERT Monte Carlo with two correlation modes: single-factor (Gaussian copula) and full correlation matrix (Cholesky)
Test method	Independent reimplementation in Python (NumPy + SciPy), mirrored line by line
Number of tests	19 (calculation engine 14 + Cholesky correlation mode 5)
Outcome	All tests passed within pre-defined tolerances
Reproducibility	Fixed seeds and explicit iteration counts; see appendix C
Status	Fit for use: suitable for substantiating risk contingencies
Prepared by	Quantil

Quantil

Quantil is a trading name of Bouman Management B.V., Barendrecht, the Netherlands | Netherlands CoC 98301764
© 2026 Quantil. Quantil is a trading name of Bouman Management B.V. All rights reserved.

Table of Contents

This table of contents updates automatically. In Word, right-click the list and choose Update Field to refresh the page numbers.

Reader's guide 3
1. Summary 3
2. Purpose and scope 3
3. Test method 4
4. Results 4
5. Testing at the scale of large projects 4
5a. Testing on a complete register (an earlier register of the same type, in EUR) 5
6. Conclusion 6
Appendix A. Algorithms and formulas 7
A.1 Beta-PERT distribution and shape parameters 7
A.2 Sampling via inverse Beta 7
A.3 Occurrence trigger (Bernoulli) 7
A.4 Correlation: single-factor Gaussian copula 8
A.5 Helper functions for the normal distribution 8
A.6 Aggregation, sorting and percentiles 8
A.7 Second correlation mode: full matrix via Cholesky 8
A.8 Pseudo-random generator (MRG32k3a) 9
Appendix B. Test set-up and measured values 10
B.1 Correlation across a spectrum of link strengths 10
B.2 Validation of the Cholesky correlation mode 11
Appendix C. Reproducibility and environment 12
C.1 Environment 12
C.2 Seeds and iteration counts 12
Appendix D. Modeling choices and their limits 13
D.1 Schedule: hybrid critical path aggregation 13
D.2 Correlation through shared factors and independent impact 13
D.3 Cost and schedule impact of the same risk 13
D.4 Shape factor λ = 4 13
D.5 Capacity and percentile method 14
D.6 Positioning and boundaries 14
Appendix E. Literature and standards 15

Reader's guide

This note is layered so that it serves the full audience: executives and clients, risk managers and cost engineers, and quants and programmers who want to be able to assess the calculation engine independently.

Main text (§1 through §6). Accessible, without formulas. Describes purpose, method, results and conclusion at the level a steering committee or investment committee needs in order to rely on the outcomes.

Appendices (A through E). Technical and complete. The exact algorithms with formulas (A), the full test set-up with tolerances and measured values (B), reproducibility and environment (C), a candid discussion of modeling choices and their limits (D), and the literature and standards the method rests on (E).

1. Summary

The calculation engine performs a Beta-PERT Monte Carlo simulation. For each risk, a probability of occurrence and a three-point estimate (minimum, most likely, maximum) for cost and schedule are processed into a probability distribution of the total project outcome. From that distribution follow the percentiles (P50 through P90), the expected value and the contingency required.

Testing was carried out by detaching the complete calculation logic from the tool and rebuilding it independently in Python with NumPy and SciPy. The outcomes of that independent reference were compared against the calculation engine, on nineteen points and against pre-defined numerical tolerances: fourteen for the calculation engine and five for the Cholesky correlation mode. All nineteen tests pass. The simulation is mathematically correctly implemented; the outcomes follow solely from the risk register entered and the parameters chosen.

Key finding for the subject-matter reader: the tool has two correlation modes that the user chooses in advance. In single-factor mode, risks are linked through shared underlying factors; the link strength L then translates into a mutual (Pearson) correlation of L^2. In Cholesky mode, the user supplies a full correlation matrix between the risk pairs, including negative correlations and separate clusters. Both modes have been validated independently; the choice between them and their limits are set out in appendix D.

2. Purpose and scope

The purpose is to establish whether the calculation engine works mathematically correctly: whether the sampling, the simulation, the percentile calculation and the correlation structure produce the intended statistical outcomes. The testing addresses the correctness of the calculation, not the substantive quality of any particular risk register.

Within scope: the derivation of the Beta-PERT shape parameters; the inverse-beta sampling; the two statistical helper functions (inverse normal distribution and normal distribution function) that the tool uses without external add-ins; the aggregation of cost (sum) and schedule (hybrid critical path plus base duration); the percentile and expected-value calculation; the pseudo-random generator (MRG32k3a) and stability across random streams; and the correlation structure.

Out of scope: the correctness of a register entered by the user, the choice of probabilities and three-point estimates, and organizational aspects of use. The reliability of a forecast remains only as good as the quality of the input.

3. Test method

The calculation engine was read line by line and the mathematical logic was rebuilt independently in a second environment (Python with NumPy and SciPy). This approach (reimplementation by an independent route and comparison of the outcomes) is the standard method for validating a numerical engine: an error in the original implementation would almost certainly manifest differently in the independent rebuild.

For every component, a testable criterion with a numerical tolerance was defined in advance. Comparison was made against (a) a known analytical formula, (b) an established reference library (SciPy), or (c) a theoretically expected statistical property. For the statistical tests, large samples (up to 200,000 draws) were used so that the properties could be established reliably. The full set-up, including seeds and measured values, is in appendices B and C.

4. Results

The table below summarizes the tests. Every component reproduces the independent reference within the pre-defined tolerance. The exact measured values and thresholds are in appendix B. Amounts in this note are stated in the currency used in the validation runs (EUR); the engine performs no currency conversion, and the mathematical results are independent of the currency label.

Component tested	Result	Verdict
Inverse normal distribution (Acklam)	max. deviation < 3×10^{-9}	**Pass**
Normal distribution function (Abramowitz-Stegun)	max. deviation < 8×10^{-8}	**Pass**
Beta-PERT shape parameters	exactly equal to the canonical PERT formula	**Pass**
Beta-PERT mode	mode reproduces the most likely value	**Pass**
Inverse-beta sampling	bounded in [min, max] and monotonic	**Pass**
Cost percentiles	monotonically increasing and above the base estimate	**Pass**
Expected value (EMV)	deviation 0.007% from the analytical EMV	**Pass**
Convergence	P80 differs by 0.03% between two seeds	**Pass**
Schedule aggregation (hybrid: critical path = sum, remainder = max, + base duration)	reproduces the hybrid aggregation exactly	**Pass**
Correlation: marginal distribution	mean 0.00 and sd 1.00 preserved	**Pass**
Correlation: link strength	measured correlation = L^2 (within 0.002)	**Pass**
Edge cases (P=0, P=100, single risk)	behavior exactly as analytically expected	**Pass**
Scale test (100 risks, 250 million, 50k iter)	deviation 0.003% from the analytical EMV	**Pass**

Note on accuracy. Deviations in the order of 10^{-8} concern the helper functions for the normal distribution and are negligible for any practical application. The statistical deviations (EMV, convergence, correlation) lie well within what is to be expected at the sample sizes used, and decrease further with more iterations.

5. Testing at the scale of large projects

To establish that the calculation engine remains reliable at the scale of large construction, infrastructure and investment projects, a loaded test was run: the maximum number of risks (100), a base estimate of EUR 250 million and 50,000 iterations. Arithmetic correctness proved not to depend on the size of the amount.

Number of risks	100 (full capacity)
Base estimate	EUR 250,000,000
Number of iterations	50,000
Simulated mean	EUR 356.0 million
Analytical expected value	EUR 356.0 million
Relative deviation	0.003%
P85 outcome	EUR 373.4 million

The simulated mean deviated by less than one hundredth of a percent from the analytical expected value and the percentiles were correctly ordered. This substantiates that the outcomes are usable for underpinning a risk contingency at a confidence level such as P85, the type of substantiation a steering committee or investment committee requires. Substantively, the method aligns with common industry practice: Beta-PERT Monte Carlo is the established approach for cost estimating and contingency determination on capital-intensive projects (see appendix E).

5a. Testing on a complete register (an earlier register of the same type, in EUR)

Alongside the component and scale tests, the full chain was tested on a complete, realistically populated register: fifty interrelated risks spread across the seven risk categories, with probability, three-point estimates for cost and schedule, and a correlation structure through shared factors. This tests the calculation engine not on a synthetic set-up but on the type of register that emerges from a facilitated risk workshop, from input to percentile outcome in a single pass.

The independent reimplementation reproduced the outcomes of the calculation engine across the entire percentile band (P10 through P90) to within 0.2%, well inside the Monte Carlo noise at the iteration count used. The core check (whether the simulated mean coincides with the analytical expected value, the sum of probability times PERT mean across all risks) passes at portfolio level.

Number of risks (open)	50, across seven categories, with shared-factor correlation
Base estimate	EUR 15.0 million
Analytical expected value (base + EMV)	EUR 17.70 million
Simulated mean	EUR 17.70 million
Relative deviation of the mean	0.015%
Percentile band (P10-P90) vs. reference	within 0.2% (well inside the Monte Carlo noise)
P85 outcome (cost)	EUR 18.79 million

The simulated mean coincides with the analytical expected value to within fifteen thousandths of a percent, and the percentiles are correctly ordered and lie above the base estimate. At the level of a complete register this confirms what the component and scale tests demonstrate at part level: the calculation engine adds no bias of its own, and the outcome follows solely from the register and the parameters chosen. The chain from input to a P85 risk contingency has therefore been tested as a whole, not only its individual parts.

6. Conclusion

On all nineteen points tested (fourteen for the calculation engine and five for the Cholesky correlation mode) the calculation engine reproduces the independent reference within the pre-defined tolerances, including at the scale of large projects, on a fully populated register and in the edge cases. The simulation is mathematically correctly implemented and the outcomes follow solely from the risk register entered and the parameters chosen. Subject to the modeling choices and preconditions in appendix D, the calculation engine is fit for substantiating risk contingencies on construction, infrastructure and investment projects.

The tests are reproducible using the seeds and parameters listed in appendix C; running them produces the same outcomes.

Appendix A. Algorithms and formulas

This appendix gives the exact mathematical content of the calculation engine, as implemented in the modules modMonteCarlo, modMonteCarloCholesky, modQuantilBetaExact and modRandom (engine v1.0). The notation follows the implementation.

A.1 Beta-PERT distribution and shape parameters

For each risk with a three-point estimate (a = minimum, m = most likely, b = maximum), a Beta distribution on the interval [a, b] is used. The shape parameters α and β follow from the standard PERT derivation with shape factor λ (default λ = 4):

```
alpha = 1 + lambda * (m – a) / (b – a)
beta = 1 + lambda * (b – m) / (b – a)

with lambda = 4 (default); a = min, m = most likely, b = max
```

With λ = 4, the expected value of the distribution reduces to the familiar PERT mean (a + 4m + b) / 6. The mode of the scaled Beta distribution coincides with m. Both properties have been tested (appendix B, tests 3 and 4). The code uses this parameterization literally (module modQuantilBetaExact, v1.2). An earlier version used the mathematically equivalent (mean, mode) form via μ = (a + λm + b)/(λ + 2); that yields identical α and β, but divides by (m − μ) and therefore by zero as soon as the three-point estimate is symmetric (m exactly midway between a and b). In that case the function fell back to the fixed value m, leaving the risk with no spread at all. The standard form printed above does not have that singularity and is therefore the one implemented; test 14 in appendix B covers this case. If b = a (no spread), m is used directly; this degeneracy has been tested separately.

A.2 Sampling via inverse Beta

Drawing from the Beta-PERT distribution is done through inverse transform sampling. For a uniform draw u ∈ (0,1), the inverse regularized incomplete Beta function I^{-1} yields the standardized draw, which is rescaled to [a, b]. The tool computes I^{-1} itself, without an external add-in: a bisection on the regularized incomplete beta CDF (module modQuantilBetaExact), at machine precision and validated against scipy.special.betaincinv. The earlier WorksheetFunction.Beta_Inv has thereby been replaced:

```
x = a + BetaInv(u, alpha, beta) * (b – a)

BetaInv = inverse regularized incomplete beta (I^-1)
 = the tool's own exact inverse (modQuantilBetaExact):
   bisection on the regularized incomplete beta CDF (Lentz continued fraction)
 = scipy.special.betaincinv in the reference
```

Because I^{-1} is monotonically increasing in u and its image lies in [0,1], x is guaranteed to be bounded in [a, b] and monotonic in u. Both were confirmed numerically over 100,000 draws.

A.3 Occurrence trigger (Bernoulli)

A risk contributes only when it occurs in that iteration. With probability p (from the register) and a uniform draw u_occ:

```
risk occurs <=> u_occ < p

cost: sum -> totalCost = totalCost + x_cost
schedule: hybrid critical path aggregation ->
  critical path (flag column AB): stack -> cpSched = cpSched + x_sched
  other risks: parallel              -> maxNonCP = max(maxNonCP, x_sched)
totalSched = baseSched + cpSched + maxNonCP   (baseSched = base duration, Project Setup)
```

Cost aggregates as a sum over all risks that occur. Schedule aggregates in hybrid fashion: risks marked as critical path (flag in column AB of the register) stack (sum), non-critical risks run in parallel

(maximum). That risk delay comes on top of the deterministic base duration baseSched (from Project Setup), so that the schedule KPI is the total duration and not only the delay (see appendix D).

A.4 Correlation: single-factor Gaussian copula

Risks can be related through a common underlying factor (driver). For each driver d, one standard normal draw z_d is made. For a risk j linked to driver d with link strength L:

```
z_j = L * z_d + sqrt(1 – L^2) * Phi^-1(rnd())
u_j = Phi(z_j) (Phi = standard normal CDF)
u_occ = u_j (occurrence via the shared factor)
u_imp = rnd() (impact drawn independently)

Phi^-1 = NormStdInv (Acklam), Phi = NormStdDist (Abramowitz-Stegun)
```

Two properties matter for the interpretation. First, the marginal distribution of z_j remains standard normal (mean 0, sd 1), so that the individual risk probabilities stay unbiased. Second: two risks sharing the same driver with link strength L have a mutual Pearson correlation of L^2 (not L), because each risk hangs on the shared factor with weight L and the product of the weights determines the shared variance. The shared factor drives the occurrence of the linked risks, so that in the simulation they tend to occur together; the impact magnitude per risk is then drawn independently from its own three-point estimate. Both points have been tested explicitly and are interpreted substantively in appendix D.

A.5 Helper functions for the normal distribution

The tool uses no external add-ins. The inverse normal CDF is implemented through Acklam's algorithm (rational approximation with separate tail and central regions, break point p = 0.02425). The normal CDF itself uses the Abramowitz-Stegun approximation 26.2.17 (five-term polynomial in t = 1/(1 + 0.2316419×|z|)). Both have been tested against SciPy as the reference (appendix B, tests 1 and 2).

A.6 Aggregation, sorting and percentiles

After N iterations, the cost and schedule outcomes are each sorted (shell sort in the tool). Percentiles are read off through nearest-rank indexing, without interpolation:

```
iterCost sorted ascending; index k = CLng(p * N)
P50 = iterCost[CLng(0.50*N)] , P80 = iterCost[CLng(0.80*N)]
P85 = iterCost[CLng(0.85*N)] , P90 = iterCost[CLng(0.90*N)]

mean = (1/N) * som(iterCost)
sd = sqrt( (som(iterCost^2) – N*mean^2) / (N-1) ) (steekproef-sd)
```

Nearest-rank without interpolation is a legitimate and widely used percentile method; at the iteration counts used (10,000 to 50,000) the difference from an interpolating method is negligible. The sample standard deviation uses N−1 in the denominator (Bessel correction).

A.7 Second correlation mode: full matrix via Cholesky

Alongside the single-factor copula (A.4), the calculation engine has a second mode in which the user supplies a full correlation matrix between the risks. The user chooses the mode in advance; single-factor remains the default. Cholesky mode handles up to 25 risks in the matrix; any risks beyond that run uncorrelated.

The approach is the standard Cholesky transformation. The correlation matrix C is decomposed into a lower triangular matrix L such that $C = L \times L^T$. Independent standard normal draws z are transformed with it into correlated draws, which are made uniform again through the normal CDF and drive the Beta-PERT sampling:

```
C = L * L^T            (Cholesky-Banachiewicz, lower triangular)
z_ind  = [Phi^-1(rnd()) , ... ]      (n independent normals)
z_corr = L * z_ind                   (correlated normals)
```

```
u_j    = Phi(z_corr[j])  ;  u_occ = u_j  ;  u_imp = rnd()
           (impact drawn independently, as in single-factor mode)
```

Repair of inconsistent input. A matrix filled in by a user is not always mathematically valid (positive definite), for example when A correlates strongly with B, B strongly with C, but A strongly opposite to C. In that case the Cholesky decomposition fails. The calculation engine catches this: a symmetric eigenvalue decomposition (cyclic Jacobi) clips the negative eigenvalues to a small positive value and rescales to a unit diagonal, which yields the nearest valid correlation matrix. The simulation therefore always continues and reports afterwards that the matrix was repaired. All helper routines (Cholesky, Jacobi, repair) are implemented in the tool itself, without external add-ins.

The essential difference from single-factor: Cholesky mode can reproduce any valid correlation structure, including negative correlations and multiple independent clusters, whereas single-factor can only express dependence through shared factors. As in single-factor mode, the correlated draw drives only the occurrence of linked risks; the impact magnitude is drawn independently from the risk's own three-point estimate (u_imp = rnd()). The testing of this is in appendix B (tests 15 through 19).

A.8 Pseudo-random generator (MRG32k3a)

The simulation draws all random numbers from a single pseudo-random generator. Since engine v1.0 that is MRG32k3a (L'Ecuyer, 1999), replacing the previously used built-in VBA Rnd. Rnd is a 24-bit generator with a period of roughly 1.7×10^{7}; on large runs (tens of thousands of iterations times dozens of risks, each with several draws) the number of values drawn approaches that period, after which the sequence would repeat. MRG32k3a has a period of roughly 3.1×10^{57} (2^{191}), passes the TestU01 test batteries and is the generator used by MATLAB, SAS and Arena among others. The tool's source of randomness is therefore of the same caliber as the reference generator (numpy.random.default_rng) against which the calculation engine was tested.

The generator, like the other helper functions, is implemented entirely within the tool, without external add-ins. MRG32k3a is constructed so that all intermediate products stay within the exact-integer limit of a Double (2^{53}); the implementation therefore uses Double arithmetic only and produces bit-identical outcomes on 32-bit and 64-bit Excel. The generator combines two multiple-recursive components:

```
comp. 1:  s12' = (1403580*s11 – 810728*s10)  mod m1, m1 = 4294967087
comp. 2:  s22' = (527612*s22 – 1370589*s20)  mod m2, m2 = 4294944443
output:   U    = ((s12' – s22') mod m1) / (m1 + 1)   in (0, 1)
norm   = 1 / (m1 + 1) = 2.328306549295728e-10
QSeed(seed): seed greater than 0 = deterministically reproducible
```

The implementation has been verified against an independent rebuild of the same algorithm: the same seed produces the same deterministic sequence, the outcomes lie strictly within the open interval (0,1) (never exactly 0 or 1, which rules out division by zero and Log(0) in the inverse normal and the Beta inverse) and the mean over one million draws is 0.4997. As a reproducibility anchor, after QSeed 42 the generator produces the first five values 0.644134970843; 0.224038350536; 0.889938362201; 0.541043519167; 0.488716078609. The built-in self-test QRand_SelfTest reproduces these values inside the tool.

Replacing the generator leaves the conclusions of appendix B untouched. The statistical properties tested there (the standard normal marginal, the L^2 correlation and the coincidence of the simulated mean with the analytical expected value) follow from the transformations, not from the specific generator, and hold for any generator of sufficient quality. A given run produces a different, statistically equivalent draw sequence with MRG32k3a than before; with a fixed seed, every run remains reproducible.

Appendix B. Test set-up and measured values

For every test, a criterion and a numerical tolerance were defined in advance. The table below gives, per test, the measured value, the threshold and the sample or iteration size used. All values are the actual outcomes of the independent reimplementation.

Test	Measured value	Threshold	Sample / set-up
1. Acklam inverse normal	2.90×10^-9	1.00×10^-6	999 points, p∈[0.001;0.999]
2. Abramowitz-Stegun CDF	7.45×10^-8	1.00×10^-7	801 points, z∈[-4;4]
3. Beta-PERT α/β	0 (exact)	1.00×10^-12	analytical
4. Beta-PERT mode	0 (exact)	1.00×10^-9	analytical
5. Inverse-beta sampling	0 (exact)	logical (true/false)	100,000 draws
6. Cost percentiles	0 (exact)	logical (true/false)	20 risks, 50,000 iter
7. Expected value (EMV)	6.94×10^-5	0.00100	20 risks, 50,000 iter
8. Convergence (2 seeds)	2.95×10^-4	0.00500	50,000 iter, seeds 11/99
9. Schedule aggregation (hybrid critical path)	0 (exact)	logical (true/false)	3 risks, 20,000 iter
10. Correlation: marginal	0.00251	0.0200	200,000 draws, L=0.7
11. Correlation: strength L²	0.00206	0.0100	150,000 draws, L=0.7
12. Edge cases	0 (exact)	logical (true/false)	5,000 to 20,000 iter
13. Scale test	3.26×10^-5	0.00100	100 risks, 50,000 iter
14. Beta-PERT with a symmetric three-point estimate	α = β = 3 (exact, 2.2×10^-15); σ 0.27% of analytical	α/β: 1.00×10^-12 σ: 1%	seed 42: 100,000 draws (the tool's MRG32k3a)

Selected measured values in detail. Acklam: max. |deviation| = 2.90×10^{-9} over 999 points. Abramowitz-Stegun: max. |deviation| = 7.45×10^{-8}. Expected value: simulated EUR 13,665,718 against analytical EUR 13,666,667 (deviation 0.007%). Convergence: P80 = 14,449,749 versus 14,454,013 across two seeds (0.03%). Correlation L = 0.7: measured Pearson 0.488 against theoretical $L^2 = 0.490$. Scale test: simulated EUR 356.0 million against analytical EUR 356.0 million (0.003%), P85 = EUR 373.4 million. Symmetric three-point estimate (100,000 / 200,000 / 300,000): α and β come out at exactly 3 (deviation 2.2×10^{-15} relative to the canonical derivation). With the tool's generator (seed 42, 100,000 draws) the simulated mean is EUR 199,968 against analytical EUR 200,000 (0.016%) and the standard deviation EUR 37,693 against analytical EUR 37,796 (0.27%), both within the Monte Carlo noise at this sample size. The distribution therefore retains its spread, where the earlier parameterization produced a point value.

B.1 Correlation across a spectrum of link strengths

To confirm that the L^2 relationship holds across the whole range and not only at a single value, the measured correlation was compared with theory for four link strengths:

Link strength L	Theory L^2	Measured correlation	Deviation
0.3	0.090	0.086	0.004
0.5	0.250	0.250	0.000
0.7	0.490	0.488	0.002
0.9	0.810	0.810	0.000

The measured correlation follows L^2 across the entire range, within the statistical noise of the sample. This confirms the derivation in appendix A.4.

B.2 Validation of the Cholesky correlation mode

The second correlation mode (A.7) was tested separately. The core routines (Cholesky decomposition, Jacobi eigenvalue decomposition and the matrix repair) were replayed line by line in the independent environment and compared with NumPy; in addition, the full chain was tested on reproducing a supplied correlation matrix and on the expected effect on dispersion.

Test	Measured value	Threshold	Verdict
15. Cholesky decomposition ($L \times L^T = C$)	max. deviation 1×10^{-16}	$< 1 \times 10^{-9}$	**Pass**
16. Jacobi eigenvalues vs NumPy	max. deviation 2×10^{-15}	$< 1 \times 10^{-9}$	**Pass**
17. Repair of an inconsistent matrix	becomes positive definite (min. eigenvalue ≥ 0)	logical	**Pass**
18. Reproduction of the correlation matrix	max. deviation 0.004 (8×8, 200k)	< 0.01	**Pass**
19. Negative and positive correlation	measured 0.800 and −0.601 (target 0.80 / −0.60)	< 0.01	**Pass**

Effect on the outcome. A control test with five risks (two of them positively correlated at 0.8 and two negatively at −0.6; 200,000 iterations, seed 42) confirms the expected behavior: the mean stays practically equal to the uncorrelated run (EUR 10.768 million against EUR 10.767 million, a difference of 0.01%), while the dispersion increases (standard deviation EUR 455,127 against EUR 439,802) and the P90 comes out higher (EUR 11.392 million against EUR 11.349 million). Positive dependence therefore thickens the tail and raises the contingency required, precisely the effect a full correlation matrix is used for.

Appendix C. Reproducibility and environment

The testing is set up to be repeatable. A third party wishing to reassess the calculation engine can reproduce the same outcomes using the details below.

C.1 Environment

- Reference implementation: Python with NumPy (fixed generator numpy.random.default_rng) and SciPy (scipy.special.betaincinv, scipy.stats.norm).
- Test source: the calculation engine modMonteCarlo (engine v1.0) from the tool, mirrored line by line, including the Acklam constants, the Abramowitz-Stegun coefficients and the shell sort.
- Basis of comparison: known analytical formulas, SciPy as an independent library, and theoretically expected statistical properties.

C.2 Seeds and iteration counts

Every statistical test uses a fixed seed, so that the outcome is deterministically reproducible:

Test	Seed / iterations
Inverse-beta sampling	seed 42: 100,000 draws
Cost percentiles / EMV	seed 2025: 50,000 iterations
Convergence	seeds 11 and 99: 50,000 iterations each
Schedule aggregation	seed 7: 20,000 iterations
Correlation (marginal and strength)	seeds 3 and L×100: 150,000 to 200,000 draws
Edge cases	seed 1: 5,000 to 20,000 iterations
Scale test	seed 777 (register), seed 123 (build): 50,000 iterations
Symmetric three-point estimate	seed 42: 100,000 draws

Small differences in the generator or the seed lead to statistically equivalent, not bit-identical outcomes; the tolerances in appendix B are calibrated accordingly.

Appendix D. Modeling choices and their limits

An honest validation states not only that the implementation is correct, but also which modeling choices were made and where their limits lie. The following choices determine how the outcomes should be read.

D.1 Schedule: hybrid critical path aggregation

Schedule impact is aggregated in hybrid fashion. Risks the user marks as critical path (flag in column AB of the register) are summed: serial, consecutive delays on the same path then stack into a longer duration. Risks without that marking run in parallel and are aggregated as the single largest impact (maximum). The sum of the critical path delay and the largest parallel delay comes on top of the deterministic base duration (baseSched, from Project Setup), so that the schedule KPI shows the total duration and not only the risk delay. This hybrid choice is deliberate: it avoids both the understatement of simply taking the maximum of serial delays and the overstatement of simply summing all parallel delays. The limit of the assumption lies in the correctness of the critical path marking: it determines which risks stack and which run in parallel, and is therefore a modeling choice made by whoever compiles the register.

D.2 Correlation through shared factors and independent impact

The user chooses in advance between single-factor and Cholesky. Each has its own field of application.

- **Single-factor.** Fast and requiring little input; suitable when risks are related mainly through a few common causes. Point of attention: the link strength L translates into a mutual correlation of L^2, so an intuitive "strong dependence" requires a higher L than one would instinctively choose (for 0.5, L ≈ 0.71 is needed).
- **Cholesky.** Full control: any desired correlation between any risk pair, including negative correlations and independent clusters that single-factor cannot express. Requires more input (a matrix) and is limited to 25 risks. An inconsistent matrix is repaired automatically (A.7).

The two modes treat impact the same way. In both, the correlation (the shared factor in single-factor mode, the full matrix in Cholesky mode) drives only the occurrence of linked risks; the impact magnitude is then drawn independently from the risk's own three-point estimate, keeping dependence in occurrence and uncertainty in magnitude cleanly separated. The net effect in both modes is that linked risks occur together more often, which widens the tail of the portfolio distribution and raises the contingency required.

D.3 Cost and schedule impact of the same risk

For each risk, the calculation engine draws one uniform value for the impact and uses it for both the cost and the schedule distribution (u_cost = u_schedule). If the cost impact of a risk that has occurred falls in the 90th percentile of its three-point estimate, the schedule impact falls in the same percentile of ITS estimate. Within a single risk, cost and time are therefore perfectly rank-dependent; between risks the draws remain independent, subject to the correlation structure from D.2.

This choice is deliberate. An event that turns out expensive (a foundation that has to be substantially heavier, a main component that arrives much later) is usually also heavy in time; independent drawing would average that dependence out and make the joint cost-time tail too thin. The limit of the assumption lies with risks where cost and time work against each other, such as accelerating at extra cost or accepting delay to save money: the calculation engine does not model that trade-off. Anyone wanting to include such a risk explicitly splits it into two register rows, one with cost impact only and one with schedule impact only.

D.4 Shape factor λ = 4

The Beta-PERT shape factor is set by default to λ = 4, the classic PERT setting that weights the most likely value more heavily than a triangular distribution would. A higher λ makes the distribution more peaked around m (less spread), a lower λ flatter. The value is adjustable; λ = 4 is the common choice and the one tested here.

D.5 Capacity and percentile method

- Capacity: the calculation engine handles up to roughly 100 risks per session. This is a design limit, not an arithmetic one: on very large programs, risks are clustered in advance or the program is split into sub-analyses.
- Percentiles via nearest-rank without interpolation. At 10,000 to 50,000 iterations the difference from an interpolating method is smaller than the remaining uncertainty; at much lower iteration counts this would become more noticeable.
- Input dependence: the calculation engine adds no assumptions of its own to the register. The reliability of a forecast is only as good as the quality of the probabilities and three-point estimates entered, hence the importance of a careful, facilitated session.

D.6 Positioning and boundaries

The calculation engine is built and validated for a specific purpose: quantifying cost and schedule uncertainty in a facilitated risk workshop and substantiating a risk contingency at a chosen confidence level. Within that purpose it does what the established method prescribes, and the correctness of that has been demonstrated in this note. For a fair expectation, it is worth stating what it is not intended for.

- **Correlation structure.** The tool offers a full correlation matrix through Cholesky mode (up to 25 risks), alongside single-factor mode. For portfolios in which far more than 25 risks each need an explicitly supplied mutual correlation, specialized packages (such as @RISK or Primavera Risk Analysis / Oracle Crystal Ball) are the appropriate instrument.
- **Schedule risk.** Schedule aggregation works at risk level (hybrid: sum over the critical path risks, maximum over the parallel risks, plus the deterministic base duration), not at activity level within a full schedule network. For a schedule risk analysis that computes the logic of a Primavera or MS Project schedule at activity level, a network-based schedule risk package is the right choice; this calculation engine complements it rather than replacing it.
- **Scale and environment.** The engine runs in Excel/VBA. That is a deliberate choice for transparency and accessibility (the user can open and follow the calculation, there is no black box) with the trade-off that execution is slower than a compiled engine and capacity sits at roughly one hundred risks per session. For portfolios with thousands of risks, or for very high iteration counts under time pressure, a compiled solution is more efficient.

In short: for risk workshops and for substantiating contingencies on construction, infrastructure and investment projects, the calculation engine is a complete and validated instrument. For activity-level schedule risk, full correlation matrices or very large portfolios, the specialized packages named above are the appropriate tool. The two can complement each other.

Appendix E. Literature and standards

The method used and the validation approach rest on established sources:

- **AACE International, Recommended Practice 41R-08:** Risk Analysis and Contingency Determination Using Range Estimating; and 57R-09 / 65R-11 for Monte Carlo based contingency determination.
- **Vose, D. (2008),** Risk Analysis: A Quantitative Guide (3rd edition), Beta-PERT distribution, inverse transform sampling and copula correlation.
- **Acklam, P. J. (2003),** An algorithm for computing the inverse normal cumulative distribution function, the inverse normal approximation the tool uses.
- **Abramowitz, M. & Stegun, I. (1964),** Handbook of Mathematical Functions, formula 26.2.17, the normal CDF approximation the tool uses.
- **Malcolm, Roseboom, Clark & Fazar (1959),** the original PERT publication; basis of the (a + 4m + b)/6 derivation.

The combination of Beta-PERT sampling with a copula correlation structure is the established approach for cost estimating and contingency determination on capital-intensive construction, infrastructure and investment projects.

Prepared by
Quantil is a trading name of Bouman Management B.V. | Netherlands CoC 98301764 | Barendrecht